UNITED STATES PATENT APPLICATION FOR:

SYSTEM AND METHOD FOR WEB APPLICATION PROPAGATION

Inventors:

Wayne M. Adams
Peter Laird
Tanya Saarva

CERTIFICATE OF MAILING BY "EXPRESS MAIL"
UNDER 37 C.F.R. §1.10

"Express Mail" mailing label number: EV327622483US
Date of Mailing: 2/17/04

I hereby certify that this correspondence is being deposited with the United States Postal Service, utilizing the "Express Mail Post Office to Addressee" service addressed to: **MAIL STOP PATENT APPLICATION, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450** and mailed on the above Date of Mailing with the above "Express Mail" mailing label number.

Tina M. Galdos (Signature)

Name: Tina M. Galdos
Signature Date: 2/17/04

SYSTEM AND METHOD FOR WEB APPLICATION PROPAGATION

Inventors:

Wayne M. Adams
Peter Laird
Tanya Saarva

COPYRIGHT NOTICE

CLAIM OF PRIORITY

[0002] This application claims priority from the following application, which is hereby incorporated by reference in its entirety:

[0003] SYSTEM AND METHOD FOR WEB APPLICATION PROPAGATION, U.S. Application No. 60/451,867, Inventors: Wayne M. Adams et al., filed on February 24, 2003. (Attorney's Docket No. BEAS-1432US0)

FIELD OF THE DISCLOSURE

[0004] The present invention disclosure relates to tools for staging web applications on a web server.

BACKGROUND

[0005] To support stable production systems, developers of enterprise application software are encouraged to deploy the software in development, staging and production systems. Each system can include one or more servers. A development system is intended for building and testing development software. A staging system is expected to be configured similarly to the production system so that various runtime aspects of the application can be tested before being deployed on the production system. A missing but

highly desirable component to complement this strategy is an automatic and systematic means for propagating the application and its resources from one system to the next.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] **Figure 1** is an illustration of an exemplary system in an embodiment of the invention.

[0007] **Figure 2** is a flow chart illustrating exemplary application component processing in an embodiment of the invention.

DETAILED DESCRIPTION

[0008] The invention is illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" embodiment in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

[0009] Application propagation refers to the process of moving an application's components and its configuration information from a source operational environment to a destination operation environment such that the application can execute in the destination environment. In one embodiment, application propagation can be performed without having to bring down the application. By way of a non-exhaustive example, application components and configuration information can include any of the following: binary executable files, J2EE (Enterprise Java) applications, .Net applications, web applications, distributed objects, libraries, configuration files, information in databases including database records, LDAP (Lightweight Directory Access Protocol) information, any type of file (e.g., binary, ASCII, image, sound, multi-media), Java Archives (JARs), XML (Extensible Markup Language) documents, HTML (Hypertext Markup Language) documents, and any other file/resource required by an application. By way of a non-limiting example, an operational environment can include one or more computing devices such as clients, servers, hand-held computers (e.g, PDAs), cellular telephones, digital music players, and/or digital cameras, and wherein the one or more computing devices can be coupled to one another via a bus, shared memory, one or more public or private computer networks, and/or other suitable means.

[0010] **Figure 1** is an illustration of an exemplary system in an embodiment of the invention. The system propagates an application's components from a source operational environment to a destination operational environment. Although this diagram depicts objects/processes as logically separate, such depiction is merely for illustrative purposes. It will be apparent to those skilled in the art that the objects/processes portrayed in this figure can be arbitrarily combined or divided into separate software, firmware and/or hardware components. Furthermore, it will also be apparent to those skilled in the art that such objects/processes, regardless of how they are combined or divided, can execute on the same computing device or can be distributed among different computing devices connected by one or more networks or other suitable communication means.

[0011] In one embodiment, the system can include a user interface (UI) **102**. The UI allows a user to interactively control the propagation process, observe its progress, and preview what changes will take place on a destination operational environment. In one embodiment, the UI may be a graphical user interface (GUI). In another embodiment, the UI can be a command line based interface. In another embodiment, the UI can include the ability to respond to sounds (e.g., voice commands), gestures, remote controls (e.g., PDAs), gestures and other suitable means. In another embodiment, the UI can include one or more of the proceeding embodiments.

[0012] In another embodiment, a process interface (PI) **116** allows another process **110** on the same or a different computing device to interactively control the system. In another embodiment, the system can be driven by commands in a batch file (not shown).

[0013] In one embodiment, the UI/PI has the capability to manually select what application components will be propagated.

[0014] In another embodiment, the UI and the PI allow a user or process to create and manipulate a set of propagation rules **104** which are used to guide automatic propagation of application components. By way of a non-limiting example, rules can specify how to handle certain situations and/or be used to explicitly include or exclude components from propagation. A difference engine **112** can propagate an application component from a source environment to a destination environment based on one or more rules within the rule set. In one embodiment, and under control of a threading

model **114**, multiple instances of the difference engine can execute simultaneously and/or in parallel. By way of a non-limiting example, the threading model can spawn a thread/process to execute an instance of the difference engine for one or more components of the application. Threads/processes can execute on the same computing device or on different computing devices. In this way, the overall time to propagate an application can be reduced.

[0015] In one embodiment, the UI/PI has the capability to provide a preview of the changes to a destination environment as a result of propagating an application. This function can be supported by the difference engine wherein the difference engine can determine what changes will be made to the destination without making those changes.

[0016] **Figure 2** is a flow chart illustrating exemplary application component processing in an embodiment of the invention. Although this figure depicts functional steps in a particular order for purposes of illustration, the process is not limited to any particular order or arrangement of steps. One skilled in the art will appreciate that the various steps portrayed in this figure could be omitted, rearranged, combined and/or adapted in various ways.

[0017] In one embodiment, the difference engine can determine how to propagate one or more application component as follows. In step **200**, it is determined whether or not there are more components to process. If there are no more components, then the process completes **202**. Otherwise, in step **204** it is determined whether or not a given component exists in the destination, but was deleted from the source environment. If this is the case, a rule can be applied in step **206**. In one embodiment, the rule can specify whether the difference engine should keep the component in the destination environment or remove it. In another embodiment, the rule can specify whether to prompt a user for a decision. In another embodiment, the rule can specify that no action should be taken. In another embodiment, the rule could initiate further analysis such as consulting an application manifest to determine whether or not the component is still needed for the application to execute.

[0018] In step **208**, it is determined whether or not a component exists in the source environment but was deleted from the destination environment. If this is the case, a rule can be applied in step **206**. In one embodiment, the rule can specify whether the difference engine should copy the component to the destination environment. In another

embodiment, the rule can specify whether to prompt a user for a decision. In another embodiment, the rule can specify that no action should be taken. In another embodiment, the rule could initiate further analysis such as consulting an application manifest to determine whether or not the component is still needed for the application to execute.

[0019] In step **212**, it is determined whether or not a component was modified in the source environment but not in the destination environment. If this is the case, a rule can be applied in step **214**. In one embodiment, the rule can specify whether the difference engine should overwrite the component in the destination environment with component from the source environment. In another embodiment, the rule can specify whether to prompt a user for a decision. In another embodiment, the rule can specify that no action should be taken. In another embodiment, the rule could initiate further analysis such as comparing the two components to determine if the differences are insignificant.

[0020] In step **216**, it is determined whether or not a component was modified in the destination environment but not in the source environment. If this is the case, a rule can be applied in step **218**. In one embodiment, the rule can specify whether the difference engine should overwrite the component in the component to the destination environment with the component from the source environment. In another embodiment, the rule can specify whether to prompt a user for a decision. In another embodiment, the rule can specify that no action should be taken. In another embodiment, the rule could initiate further analysis such as comparing the two components to determine if the differences are insignificant.

[0021] In step **220**, it is determined whether or not a component is new in the source environment but not in the destination environment. If this is the case, a rule can be applied in step **222**. In one embodiment, the rule can specify whether the difference engine should copy the new component to the destination environment. In another embodiment, the rule can specify whether to prompt a user for a decision. In another embodiment, the rule can specify that no action should be taken. In another embodiment, the rule could initiate further analysis such as comparing the two components to determine if the differences are insignificant.

[0022] In step **224**, it is determined whether or not a component is new in the destination environment but not in the source environment. If this is the case, a rule can be applied in step **226**. In one embodiment, the rule can specify whether the difference

engine should remove the component from the destination environment. In another embodiment, the rule can specify whether to prompt a user for a decision. In another embodiment, the rule can specify that no action should be taken. In another embodiment, the rule could initiate further analysis such as comparing the two components to determine if the differences are insignificant.

[0023] One embodiment may be implemented using a conventional general purpose or a specialized digital computer or microprocessor(s) programmed according to the teachings of the present disclosure, as will be apparent to those skilled in the computer art. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art. The invention may also be implemented by the preparation of integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

[0024] One embodiment includes a computer program product which is a storage medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the features presented herein. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[0025] Stored on any one of the computer readable medium (media), the present invention includes software for controlling both the hardware of the general purpose/specialized computer or microprocessor, and for enabling the computer or microprocessor to interact with a human user or other mechanism utilizing the results of the present invention. Such software may include, but is not limited to, device drivers, operating systems, execution environments/containers, and user applications.

[0026] The foregoing description of the preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. Embodiments were chosen and described in order to best describe the principles of the

invention and its practical application, thereby enabling others skilled in the art to understand the invention, the various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.